

UNITED STATES PATENT APPLICATION

for

A Prioritized Address Decoder

Inventors:

Douglas R. Moran

Satish Acharya

Zohar Bogin

Sean G. Galloway

Prepared by:

BLAKELY SOKOLOFF TAYLOR & ZAFMAN LLP  
12400 Wilshire Boulevard  
Los Angeles, CA 90025-1030  
(408) 720-8300

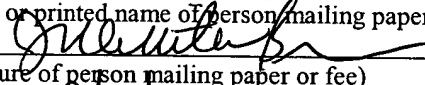
Attorney's Docket No.: 042390.P17504

"Express Mail" mailing label number: EV336590302 US

Date of Deposit: 9/19/03

I hereby certify that I am causing this paper or fee to be deposited with the United States Postal Service "Express Mail Post Office to Addressee" service on the date indicated above and that this paper or fee has been addressed to the Commissioner for Patents, P.O. Box 1450, Alexandria, VA 22313-1450.

JUANITA BRISCOE  
(Typed or printed name of person mailing paper or fee)

  
(Signature of person mailing paper or fee)

9/19/03  
(Date signed)

## **A Prioritized Address Decoder**

### **FIELD OF INVENTION**

**[0001]** The present invention relates to computer systems, and more particularly, to data security in a computer system.

### **BACKGROUND**

**[0002]** In a typical computer system, a memory controller or a memory controller hub (MCH) routes data in between various devices within the computer system, such as, a processor, a main memory, a graphics chip, a peripheral device, etc. Some of the devices of the computer system are referred to as trusted agents because it is safe to send secured data to these devices. For example, the Central Processing Unit (CPU) is a trusted agent in one computer system. The remaining devices are referred to as non-trusted agents.

**[0003]** The MCH in the computer system allows software to allocate memory space in a memory map for various devices in the computer system. When the computer system is initialized, the basic input/output software (BIOS) programs a set of configuration registers in the MCH to define a memory map for the computer system.

**[0004]** Figure 1 shows an example of the memory map 100. The bottom portion 120 of the memory map 100 is assigned to the main memory of the computer system. Memory portions 111, 113, and 115 are respectively assigned to devices A, B, and C of the computer system. Usually, the portions of the memory map for the devices do not overlap with each other or with the portion for the main memory. To route data within the computer system, the MCH decodes the destination address of the data to determine

in which device's address range the destination address falls into. Then the MCH routes the data to that device.

**[0005]** An existing address decoder in a MCH is shown in Figure 2. The address decoder includes a number of address comparators 210 connected in parallel. Each comparator compares the destination address of the data with an address range of a device within the system. The values of `cfg_bitsA` 203, `cfg_bitsB` 205, and `cfg_bitsC` 207 represent the address ranges of devices A, B, and C respectively. The address range of the main memory is represented by `cfg_bitsN` 209. If the destination address falls within the address range of a device, the corresponding comparator outputs a signal to enable the MCH to route the data to the device. Since each comparator is independent of the other comparators, the same data may be written to multiple devices when the address ranges of the multiple devices overlap with each other and the destination address falls into the overlapped range. For example, referring to the memory map 300 in Figure 3, the address range of device C 315 overlaps with the address range of the main memory 320. When the destination address of the data falls within the overlapping address range 315, the data is written to both the main memory and device C.

**[0006]** Some software may be used to exploit the fact that data is sent to multiple locations when address ranges overlap in order to steal secured data from the computer system. For example, the software reprograms the address range of a non-trusted agent, e.g., a peripheral device, to overlap with the address range of a trusted agent. When the trusted agent accesses the secured data, the non-trusted agent receives the secured data as well if the destination address of the secured data falls into the address range shared by both the trusted agent and the non-trusted agent. However, it is impractical to bar

reprogramming of the address ranges of peripheral devices because other legitimately operating software applications may reprogram the address ranges from time to time.

## BRIEF DESCRIPTION OF THE DRAWINGS

[0007] The present invention will be understood more fully from the detailed description that follows and from the accompanying drawings, which however, should not be taken to limit the appended claims to the specific embodiments shown, but are for explanation and understanding only.

[0008] Figure 1 shows an example of a memory map.

[0009] Figure 2 shows an existing address decoder.

[0010] Figure 3 shows another example of a memory map.

[0011] Figure 4A shows one embodiment of a prioritized address decoder.

[0012] Figure 4B shows an alternate embodiment of a prioritized address decoder.

[0013] Figure 4C shows one embodiment of a prioritized address decoder.

[0014] Figure 5 shows a flow diagram of one embodiment of a process for routing data in a computer system.

[0015] Figure 6 shows an exemplary embodiment of a computer system.

## DETAILED DESCRIPTION

**[0016]** In the following description, numerous specific details are set forth. However, it is understood that embodiments of the invention may be practiced without these specific details. In other instances, well-known circuits, structures, and techniques have not been shown in detail in order not to obscure the understanding of this description.

**[0017]** Figure 4A shows one embodiment of a prioritized address decoder 400. The prioritized address decoder 400 may be part of a MCH in a computer system. In one embodiment, the decoder 400 includes 3 address comparators 410-430 and an OR gate 440. The address comparators 410-430 compare an input address 401 with `cfg_bitsA` 403, `cfg_bitsB` 405, and `cfg_bitsC` 407, respectively. In one embodiment, the input address 401 is the destination address of the data to be sent to a device of the computer system. In one embodiment, `cfg_bitsA` 403, `cfg_bitsB` 405, and `cfg_bitsC` 407 correspond respectively to the address ranges of devices A, B, and C within the computer system. Examples of devices A, B, and C include the main memory, the graphics chip, etc. In one embodiment, `cfg_bitsA` 403, `cfg_bitsB` 405, and `cfg_bitsC` 407 are stored in a number of configuration registers in, or accessible by, the MCH during configuration of the computer system. In one embodiment, the values of `cfg_bitsA` 403, `cfg_bitsB` 405, and `cfg_bitsC` 407 may be modified by software after configuration.

**[0018]** Referring to Figure 4A, the output of the comparator 410 is coupled to a select input of the comparators 420 and 430. If the input address 401 falls within the address range corresponding to device A, then the output of the comparator 410, `DestinationA` 493, goes high to allow the data to go to device A. Also, the output of the

comparator 410 at high disables the remaining comparators 420 and 430 so that the data would not be sent to device B or device C.

**[0019]** In one embodiment, if the input address 401 does not fall within the address range of device A, the output of the comparator 410, DestinationA 493, goes low to prevent the data from going to device A and enables the comparator 420. When the comparator 420 is enabled, the comparator 420 compares the input address 401 with cfg\_bitsB 405 and determines whether the input address 401 is within the address range of device B. If the input address 401 is within the address range of device B, the output of the comparator 420, DestinationB 495, goes high to allow the data to go to device B. DestinationB 495 also goes to the comparator 430 via the OR gate 440 to disable the comparator 430.

**[0020]** In one embodiment, the outputs of the comparators 410 and 420 are coupled to inputs of the OR gate 440. If the input address is not within the address range of device A or the address range of device B, then the outputs of the comparators 410 and 420 go low, i.e., both DestinationA 493 and DestinationB 495 go low. DestinationA 493 and DestinationB 495 are input to the OR gate 440, and therefore, the output of the OR gate 440 goes low to enable the comparator 430. The comparator 430 compares the input address 401 with cfg\_bitsC 407 to determine whether the input address 401 is within the address range of device C. If so, the output of the comparator 430, DestinationC 497, goes high to allow the data to go to device C.

**[0021]** In an alternate embodiment, the prioritized address decoder includes a different number of comparators, such as, for example, 2, 4, 5, etc., that may depend on the number of devices in the system that have associated address ranges. In one

embodiment, there is one comparator for each device in the computer system. Figure 4B shows one embodiment of a prioritized address decoder 490. Referring to Figure 4B, the comparators 492 are arranged in series with the OR gates 494 coupled in between the comparators 492. The comparators 492 compare the input address 491 to address ranges corresponding to devices in the computer system one by one. When one of the comparators 492 determines that the input address 491 is within the address range associated with the comparator, the comparator disables the remaining comparators in the series. For example, in one embodiment, the decoder includes  $N$  comparators arranged in a series. When the  $k$ th comparator determines that the input address 491 is within the address range associated with the  $k$ th comparator, the  $(k+1)$ th through  $N$ th comparators will be disabled. It should be apparent to one of ordinary skill in the art that the logic configuration disclosed can be extended to any number of comparators. The embodiments shown are merely for illustrating the concept, and thus, these embodiments should not be construed to limit the appending claims to any particular number of comparators.

**[0022]** In one embodiment, the comparators are arranged in a sequence such that the comparators assigned to the trusted agents are enabled before the comparators assigned to the non-trusted agents. Such arrangement prevents the non-trusted agents with an address range overlapping the address range of a trusted agent from accessing secured data that is to be sent to the trusted agent. It is because the comparator assigned to the trusted agent disables the comparator assigned to the non-trusted agent when the destination address of the data falls within the address range of the trusted agent. For example, referring back to Figure 4A, suppose device A is a trusted agent and device B is



a non-trusted agent, where the address range of device B overlaps with the address range of device A at the address 401. The comparator 410 checks the address 401 and generates an output to allow the data to go to device A and to disable the remaining comparators 420 and 430. Since the comparator 420, which is assigned to device B, is disabled, the data is not allowed to be sent to device B. Therefore, the prioritized address decoder 400 prevents device B from stealing the secured data from the computer system.

**[0023]** Figure 4C shows an alternate embodiment of a prioritized address decoder 450. Decoder 450 includes address comparators 412, 422, and 432, AND gates 453 and 457, and inverters 451 and 455. The address comparators 410-430 compare an input address 401 with `cfg_bitsA` 403, `cfg_bitsB` 405, and `cfg_bitsC` 407, respectively. Each of `cfg_bitsA` 403, `cfg_bitsB` 405, and `cfg_bitsC` 407 is associated with an address range of a device in a computer system. The output of the comparator 410 is DestinationA 493, which is input to the inverter 451. The output of the inverter 451 and the output of the comparator 422 are input to the AND gate 453. The output of the comparator 422 is also input to the inverter 455. The output of the inverter 455, the output of the inverter 451, and the output of the comparator 432 are input to the AND gate 457. The outputs of the AND gates 453 and 457 are DestinationB 495 and DestinationC 497, respectively. DestinationA 493, DestinationB 495, and DestinationC 497 allow data to be sent to the devices having address ranges associated with `cfg_bitsA` 403, `cfg_bitsB` 405, and `cfg_bitsC` 407, respectively.

**[0024]** In one embodiment, a comparator outputs a signal at high level and allows data to be sent to the device associated with the address range when the input address 401 falls within the associated address range of a comparator. For example, if input address

401 falls within the address range associated with `cfg_bitsA` 403, comparator 412 outputs a signal at high level to allow the data to be sent to the device associated with `cfg_bitsA` 403. The output of comparator 412 is input via the inverter 451 to the AND gates 453 and 457. The inverter 451 inverts the output of comparator 412 from a high level to a low level, and therefore, forcing the outputs of both AND gates 453 and 457, i.e., DestinationB 495 and DestinationC 497, respectively, to be at low level, regardless of the other inputs to the AND gates 453 and 457. Therefore, the data would be sent to only the device associated with `cfg_bitsA` 403, not the devices associated with `cfg_bitsB` 405 and `cfg_bitsC` 407. One should appreciate that the embodiments described above are merely for illustrating the concept. Other embodiments may include different logic circuitries or configuration without going beyond the scope and boundary of the appended claims.

**[0025]** Figure 5 shows one embodiment of a process for routing data to a device within a computer system. The process is performed by processing logic that may comprise hardware (e.g., circuitry, dedicated logic, etc.), software (such as is run on a general purpose computer system or a dedicated machine), or a combination of both. Referring to Figure 5, a device is referred to as a trusted agent if it is safe to send secured data to the device. Otherwise, the device is referred to as a non-trusted agent. Processing logic determines whether the destination address of the data is within the address range of a trusted agent (processing block 520). If the destination address of the data is within the address range of the trusted agent, processing logic sends the data to the trusted agent and the process ends (processing block 529). Otherwise, processing logic determines whether all the trusted agents in the system have been checked (processing block 525). If there is at least one trusted agent not checked yet, processing logic repeats processing

block 520 to check the remaining trusted agent(s). If all trusted agents have been checked, then processing logic moves on to check the non-trusted agents.

**[0026]** For a non-trusted agent, processing logic determines whether the destination address is within the address range of the non-trusted agent (processing block 530). If the destination address is within the address range of the non-trusted agent, processing logic sends the data to the non-trusted agent and the process ends (processing block 539). Otherwise, processing logic determines whether there is any non-trusted agent not checked yet (processing block 535). If there is a non-trusted agent not checked yet, processing logic repeats processing block 530 on the non-trusted agent until all non-trusted agents have been checked. If the destination address does not fall within the address range of any trusted or non-trusted agent, then processing logic flags an error (processing block 540).

**[0027]** Since processing logic checks all trusted agents before checking any non-trusted agent and stops looking for another agent when processing logic finds a trusted agent having an address range encompassing the destination address of the data, the data is not sent to a non-trusted agent even if the destination address is also within the address range of the non-trusted agent. Such address decoding mechanism prevents the non-trusted agent with an address range overlapping the address range of a trusted agent from accessing secured data going to the trusted agent.

**[0028]** Figure 6 shows an exemplary embodiment of a computer system 600. The system 600 includes a processor 610, a MCH 620, a main memory 630, and a number of peripheral devices 640. In one embodiment, processor 610 includes a microprocessor, but is not limited to a microprocessor, such as, for example, Pentium<sup>®</sup>,

Itanium<sup>®</sup>, PowerPC<sup>®</sup>, etc. Processor 610 is coupled to main memory 630. In one embodiment, main memory 630 includes a random access memory (RAM), or other dynamic storage device, such as, for example, a dynamic random access memory (DRAM), to store data and instructions to be executed by processor 610. The data and instructions are routed between processor 610, main memory 630, and other peripheral devices 640 via MCH 620.

**[0029]** In one embodiment, MCH 620 includes a priority address decoder 622 and a set of configuration registers 624 to route data between the devices of computer system 600. Some of the devices are referred to as trusted agents because it is safe to send secured data to these devices. The remaining devices are referred to as non-trusted agents. For example, in one embodiment, main memory 630, processor 610, and device A are trusted agents, while device B and device C are non-trusted agents.

**[0030]** To prevent routing secured data to non-trusted agents, MCH 620 checks the destination address of the secured data with the priority address decoder 622. In one embodiment, the address ranges of both the trusted and non-trusted agents are stored in the configuration registers 624. In one embodiment, the configuration registers 624 are set during configuration of various devices of the computer system 600. The contents of the configuration registers 624 may be modified during execution of certain software applications. In one embodiment, the configuration registers 624 are locked during a trusted mode to prevent unauthorized modification of the contents of the registers 624.

**[0031]** In one embodiment, the priority address decoder 622 checks the address ranges of the trusted agents one by one. In one embodiment, the priority address decoder 622 includes one comparator for each device in the computer system to determine

whether the destination address of the data falls within the address range of the device. The comparators may be arranged in a sequence such that all comparators corresponding to trusted agents are before the comparators for non-trusted agents. In one embodiment, when the priority address decoder 622 identifies the trusted agent with an address range encompassing the destination address, the corresponding comparator outputs a signal to disable the other comparators such that the secured data is allowed to go to only the trusted agent. When the decoder 622 determines that the destination address is not within the address range of any of the trusted agents, the decoder 622 checks the non-trusted agents. Hence, the decoder 622 prevents the secured data from going to a non-trusted agent with an address range overlapping the address range of a trusted agent.

**[0032]** Note that any or all of the devices of computer system 600 and associated hardware may be used in various embodiments of the present invention. However, it can be appreciated that other configurations of the computer system may include some or all of the devices.

**[0033]** The foregoing discussion merely describes some exemplary embodiments of the present invention. One skilled in the art will readily recognize from such discussion, the accompanying drawings and the claims that various modifications can be made without departing from the spirit and scope of the appended claims. The description is thus to be regarded as illustrative instead of limiting.